AMENDMENT

In the Specification

Please amend the specification as follows:

Please replace the abstract with the following paragraph:

A method and system for storing and accessing firmware that is distributed across both local and remote storage devices. This capability is facilitated by a standard software abstraction for a firmware storage device, known as a Firmware Volume (FV), which enables platform firmware to be stored in a variety of types of devices, including remote storage devices that may be access via a network. Under this distributed firmware storage architecture, platform firmware code may be written in a manner such that only the early memory initiation code and code necessary to produce or access to an FV exists in the local system ROM. All other firmware components, including device initialization and OS bootstrap code, may be located remotely. As a result, the firmware code, such as a system's BIOS, may be updated by ~~simple~~ simply updating the portion of the firmware that is stored in the firmware volume.

Please replace the paragraph beginning at page 1, line 9 with the following paragraph:

Computer platform firmware is used during initialization of computer systems to verify system integrity and configuration. It also generally provides the basic low-level interface between hardware and software components of those computer systems, enabling specific hardware functions to be implemented via execution of higher-level software instructions ~~contains~~ contained in computer programs that run on the computer systems. In computers, a primary portion of this firmware is known as the Basic Input/Output System (BIOS) code of a computer system. The BIOS code comprises a set of permanently recorded (or semi-permanently recorded in the

case of systems that use flash BIOS) software routines that provides the system with its fundamental operational characteristics, including instructions telling the computer how to test itself when it is turned on, and how to determine the configurations for various of built-in components and add-on peripherals.

Please replace the paragraph beginning at page 4, line 17 with the following paragraph:

In a conventional computer system, the BIOS starts to work as soon as a system is turned on. For example, when modern Intel microprocessors (*e.g.*, Pentium III, IV) start to work, they immediately set themselves up in real mode and look at a special memory location ~~thatis exactly 16~~ that holds a jump instruction that redirects the processor to begin execution of code at another address corresponding to where the base portion of the BIOS is actually stored, such as ROM chip 12 in FIGURE 1.

Please replace the paragraph beginning at page 6, line 8 with the following paragraph:

In accordance with this distributed platform firmware storage architecture, BIOS code may be written in a manner such that only the early memory initiation code and code necessary to produce or access to an FV exists in the local system ROM. All other firmware components, including device initialization and OS bootstrap code, may be located remotely. As a result, the BIOS code may be updated by ~~simple~~ simply updating the remote portion of the firmware. For instance, if the remote portion of the BIOS code is stored on a network firmware server, changing such code on the firmware server will effectively change the platform firmware for all computers that are configured to access a portion of their BIOS code from the firmware server.

Please replace the paragraph beginning at page 8, line 21 with the following paragraph:

Next, in a block 62, the firmware code that is to be uploaded to the computer system is authenticated and its integrity is verified. In one embodiment, the firmware volume comprises a firmware file system (FSS) that includes one or more firmware files, each containing one or more segments. Each segment contains a set of firmware code that is suitable for one or more particular computer systems. Various mechanisms can be employed to ensure the authenticity of each firmware file, as well as the integrity of the file. For example, a digital signature may be encoded into a header for the firmware file or segment, whereby it can be checked against a digital ~~signal~~ signature known to the computer system. If the digital signatures do not match, the firmware code is no valid to upload.

Please replace the paragraph beginning at page 9, line 4 with the following paragraph:

Examples of integrity checks include file state checks and checksum operations. As an example of a file state check, each file may contain embedded integrity information, such that if a failure occurs during a file alteration (*i.e.*, create, update, or delete) operation or due to a storage device failure, the file is flagged to indicate that it is invalid. This may be implemented by ~~includinga~~ including state bits in the file's header, whereby one or more state bits are set during one of the foregoing file alteration operations, and cleared after the operation completes. If the file is found to have appropriate state bits set, it is known that the create, update, or delete operation was not completed successfully, and that the file is invalid. A checksum operation may be performed on the file as a whole, or various portions of the file. For instance, a checksum value may be used to verify the integrity of a file's

header and/or main body. Checksum operations are well-known in the art; accordingly further details for performing such are not disclosed herein.

Please replace the two paragraphs beginning at page 13, line 13 with the following paragraphs (these paragraphs contain content corresponding to the overlapping words on original page 13, lines 20-23):

With reference to FIGURE 7, in one embodiment the foregoing firmware volume access mechanism is implemented as follows. During execution of early firmware 22, one or more firmware volume drivers 76 are loaded from a local firmware device 28 and executed. Execution of each firmware driver causes one or more firmware volume protocol instances 78 to be published, as depicted by FV protocol instances 78A, 78B, 78C, and 78D. Each FV protocol instance provides an abstracted interface that enables consumers of firmware to access that firmware from a corresponding firmware volume. By ~~producing~~ publishing these abstracted interfaces, the firmware volumes corresponding to those interfaces are made visible to the system.

As shown in FIGURE 8, additional firmware drivers can be loaded by firmware that has been previously retrieved from local and/or remote firmware volumes. These firmware drivers can then publish additional firmware volume protocol instances, which can be used to access additional firmware volumes. For example, during execution of early firmware 22, a firmware volume driver 76E is loaded. Firmware volume driver 76 then publishes a firmware volume protocol instance 78E that enables access to a remote firmware volume 32E. A firmware file is retrieved from remote firmware volume 32E and is loaded and executed, causing a firmware volume driver 76 to be loaded. Firmware volume driver 76 then publishes a firmware volume protocol instance 78F that enables access to a remote

firmware volume 32F. This process can be extended in a similar chained manner as many levels as desired.